Multiple Linear Regression of Hourly Bike Rentals

Anthony J. Coots

College of Information Technology, Western Governors University

D214: Data Analytics Graduate Capstone

Dr. William Sewell

June 27, 2024

Dedication

This graduate capstone study is dedicated to my family, friends, and mentors who have supported me throughout my journey. Individuals I would not be here without.

ACKNOWLEDGEMENTS

This study would not have been possible without the support and encouragement from a multitude of individuals whose contributions have been priceless. I extend my deepest gratitude to my parents, whose unwavering support has been my cornerstone. I am deeply thankful to Dr. *Ehlers*, whose passion for teaching enriches the student experience. My colleagues *Bill* and *Tanner* have been instrumental in supporting both my academic and professional pursuits and endeavors.

I must also acknowledge my friends – *Sam*, *Eric*, *Elizabeth*, and hundreds of others too numerous to list – who have provided endless inspiration and support throughout my life. Additionally, I am grateful to all the authors and researchers whose work have guided and informed this study.

Multiple Linear Regression of Hourly Bike Rentals

This study's contribution to the field of Data Analytics and the MSDA program at Western Governors University is to create a predictive Multiple Linear Regression (MLR) model capable of predicting hourly bike rental counts from environmental and seasonal factors such as month, temperature, registered users, and more. A similar study finds that among features including 'temperature' and 'hours of the day,' certain variables 'exhibited linear relationships' to predict the number of bikes rented in Seoul, South Korea (Karunanithi et al., 2024).

This study aims to create a MLR model that predicts hourly bike rentals based on several environmental and season factors utilizing Ordinary Least Squares (OLS). OLS is a method that determines the coefficients to effectively weigh each statistically significant variable used for modeling (Sahu, 2023) and for its efficiency in estimating where the independent variables are not perfectly multicollinear.

Kashyap (2021), published by the International Journal of Innovative Science and Research Technology, also demonstrated the effectiveness of using linear regression for bike sharing demand. The study detailed a linear regression equation backed by a fair-fit R2 metric of nearly 0.6, utilizing factors such as functioning hour and wind speed. This study, building on Kashyap's work, sets a clear benchmark to evaluate the model's effectiveness, aiming for an R2-score of 0.6 or greater.

Research Question

The study starts with a research question based on a provided market research dataset for a fit between environmental and seasonal factors predicting an hourly count of bike rentals. Can a MLR model be constructed on the market research dataset?

Null Hypothesis, H_0:

A predictive MLR model cannot be constructed on the market research dataset.

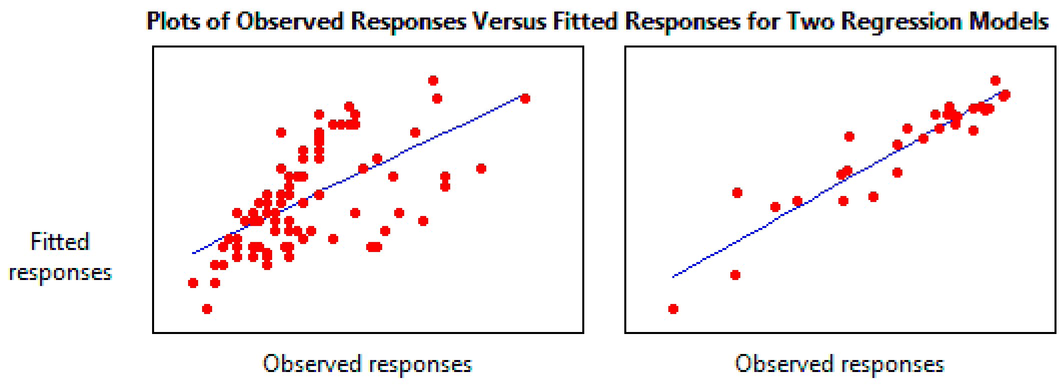
Alternate Hypothesis, H_a:

A predictive MLR model can be constructed on the market research dataset with accuracy greater than or equal to 60%

The choice of 60% accuracy measured with the R2 metric is based on benchmarking against Kashyap's (2021) findings, which demonstrated a similar model achieving this threshold, making it the target for reliable predictions in this context. Other values, such as mean squared error (MSE) and F-statistic, will be used in the study to evaluate the model's accuracy and effectiveness.

Figure 1

Graphical Representation of R-Squared.



Note. By Minitab Editor, 2013. R^2 is a key statistical measure to back an MLR model's accuracy. The metric, ranging from 0-100%, explains the variability of the response variable data of the associated mean.

Data Collection

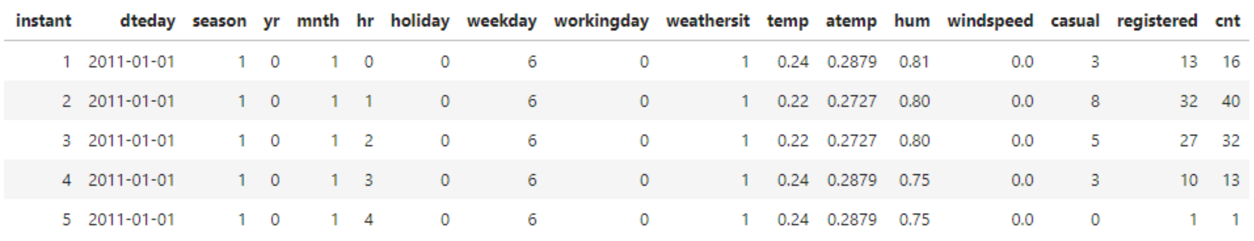
The dataset utilized in this study is publicly available through UC Irvine's Machine Learning Repository. It has 17,379 entries, excluding column headers, and was last updated/donated on December 19, 2013. This dataset is functional, given its relevant variables of environmental and seasonal factors, which are present in similar studies.

Dataset

Variable	Data Type	Independent/Dependent
instant	Categorical	Independent
dteday	Categorical	Independent
season	Categorical	Independent
yr	Categorical	Independent
mnth	Categorical	Independent
hr	Categorical	Independent
holiday	Categorical	Independent
weekend	Categorical	Independent
workingday	Categorical	Independent
weathersit	Categorical	Independent
temp	Continuous	Independent
atemp	Continuous	Independent
hum	Continuous	Independent
windspeed	Continuous	Independent
casual	Continuous	Independent
registered	Continuous	Independent
cnt	Continuous	Dependent

Figure 2

Hourly data DataFrame.



instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
1	2011-01-01	1	0	1	0	0	6	0	1	0.24	0.2879	0.81	0.0	3	13	16
2	2011-01-01	1	0	1	1	0	6	0	1	0.22	0.2727	0.80	0.0	8	32	40
3	2011-01-01	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	0.0	5	27	32
4	2011-01-01	1	0	1	3	0	6	0	1	0.24	0.2879	0.75	0.0	3	10	13
5	2011-01-01	1	0	1	4	0	6	0	1	0.24	0.2879	0.75	0.0	0	1	1

Note. The first 5 rows of the dataset loaded in a Markdown/Python environment.

Advantage

As denoted on the dataset-specific website from the UC Irvine Machine Learning Repository, the dataset is free of missing values. The whole dataset is a vital advantage of the data-gathering methodology since missing values ‘make the dataset inconsistent and unable to work on’ (Jain & Banka, 2024). Only after adequately treating the data, where addressing missing values is an important step, can the next step in the analysis process begin.

Disadvantage

A limitation or disadvantage of the dataset is the locale. The dataset depends on the Washington D.C. area, where relative factors like location and population differ globally (Eren & Uz, 2019), which would influence the use of rental bikes differently, are not in this analysis due to a lack of representation in the dataset.

Challenges

One challenge of appropriately assessing the dataset is choosing the correct CSV file. Upon download, the dataset comes with an hourly and daily dataset, and it is important to pay close attention to the latter, as the data represented in this study is from the hourly dataset and does not include the daily dataset. Otherwise the collection of data was free of challenges.

Data Extraction and Preparation

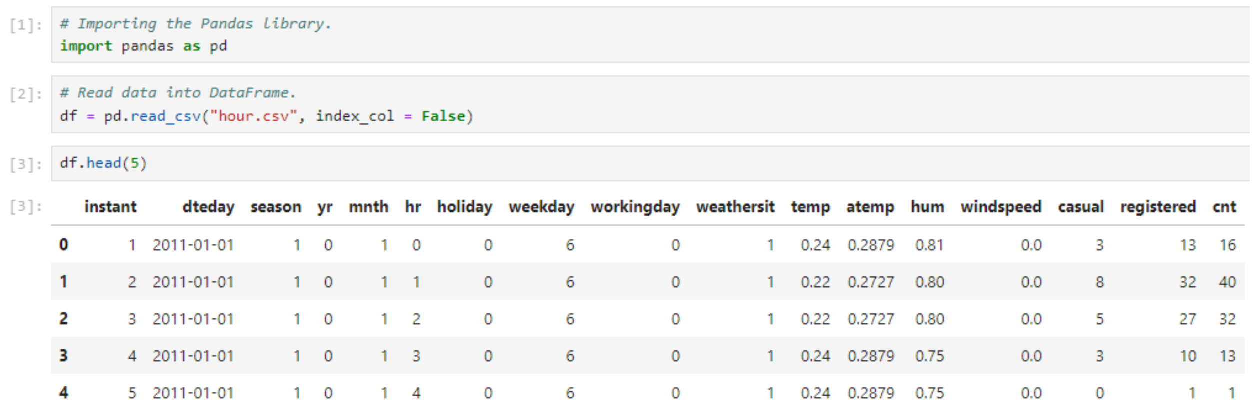
The study utilizes Python, a programming language tuned for data science, to create a MLR model for the market research dataset. Each of the data-extraction and -preparation steps are as follows.

Data Extraction

The dataset comes in a CSV file and will first be placed into a working directory to load the dataset into a Pandas DataFrame. *Pandas* is a Python library that allows for reading CSV files into a data structure ready for following data preparation and manipulation steps.

Figure 3

Data extraction, CSV to DataFrame.



```
[1]: # Importing the Pandas library.
     import pandas as pd

[2]: # Read data into DataFrame.
     df = pd.read_csv("hour.csv", index_col = False)

[3]: df.head(5)

[3]:    instant      dteday  season  yr  mnth  hr  holiday  weekday  workingday  weathersit  temp   atemp   hum  windspeed  casual  registered  cnt
     0        1  2011-01-01       1   0     1   0        0        6           0           1  0.24  0.2879  0.81        0.0       3          13   16
     1        2  2011-01-01       1   0     1   1        0        6           0           1  0.22  0.2727  0.80        0.0       8          32   40
     2        3  2011-01-01       1   0     1   2        0        6           0           1  0.22  0.2727  0.80        0.0       5          27   32
     3        4  2011-01-01       1   0     1   3        0        6           0           1  0.24  0.2879  0.75        0.0       3          10   13
     4        5  2011-01-01       1   0     1   4        0        6           0           1  0.24  0.2879  0.75        0.0       0           1    1
```

Note. Importing Pandas, extracting the data from a CSV file to a DataFrame.

Data Preparation

The detection and handling of missing values and outliers are crucial, as they can significantly influence the MLR model and decrease the strength of these models (Jain & Banka, 2024). Since no missing values exist, the next step involves detecting and imputing outliers appropriately.

Identifying which variables, independent or dependent, categorical or continuous, should be treated for outliers is critical. This study removes and imputes outliers as a preliminary measure for correspondence in assisting model performance. Consistency, a much-required aspect of the dataset before training and testing an MLR model, requires the same treatment for the dependent variable.

Handling outliers in categorical features differs fundamentally from continuous variables. For example, the dataset captures hourly data of bike rentals from the first day of 2011 until the last day of 2012, along with a variable indicating whether or not the date is a holiday from the Washington D.C. human resources holiday schedule. The holiday variable is binary and represents whether or not the date was a holiday. Since there are significantly fewer holidays than non-holiday dates and thus hours, the nature of this categorical variable would "obviously" present outliers. However, to preserve contextual importance or proper statistical representation, these variables will not be treated as "outliers" as outliers for categorical data often do not reference a consistent or agreed-upon methodology and rely entirely on the context of the variable.

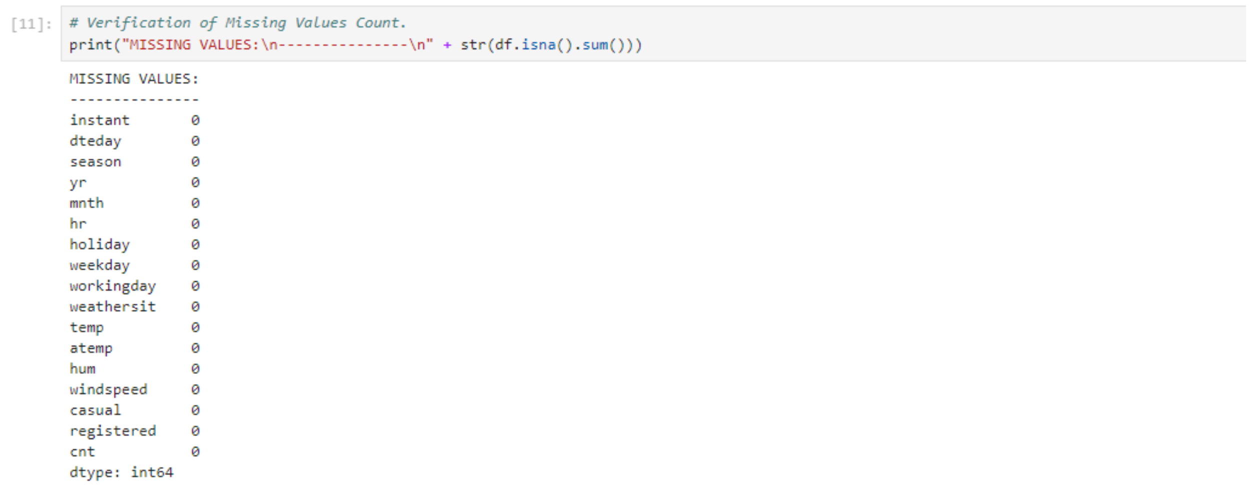
The variables' datatypes should then be assessed to provide the model with data and the respective format required for analysis. Most, if not all, linear regression models (Python or not) require a numerical format for interpretation. Categorical variables, such as Boolean, must be translated to the appropriate numerical representation of the Boolean value, where one represents true and zero represents false.

Removing variables that are not necessary for analysis helps ease the analysis step in the study. Variables such as 'dteday', which represents the date, are represented in other variables in an ordinal fashion, such as 'mnth' and 'yr.' Therefore, it is better to remove this variable as it would otherwise be omitted upon analysis primarily due to its string nature. It is not applicable to be translated into a numerical format. The variables removed are 'instant' and 'dteday' as they only represent the instance identifier and date, and can be done so with Pandas.

Finally, variables should be scaled to help align their influence in the model with other independent variables. This step standardizes all independent variables to have a mean of zero and a standard deviation of one to ensure that the regression coefficients reflect the variables equally.

Figure 4

Programmatic output to assess missing values in the dataset.



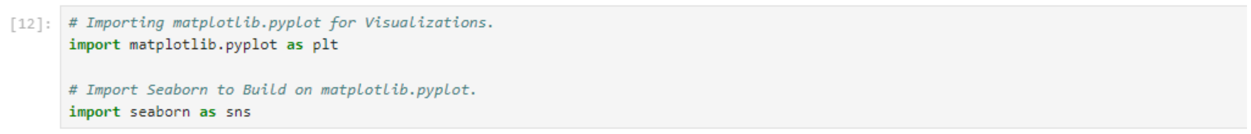
```
[11]: # Verification of Missing Values Count.
      print("MISSING VALUES:\n---------------\n" + str(df.isna().sum()))
```

```
MISSING VALUES:
---------------
instant       0
dteday        0
season        0
yr            0
mnth          0
hr            0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
casual        0
registered    0
cnt           0
dtype: int64
```

Note. Verification of no missing values by detection per feature.

Figure 5

Importing necessary libraries/modules for visualization.



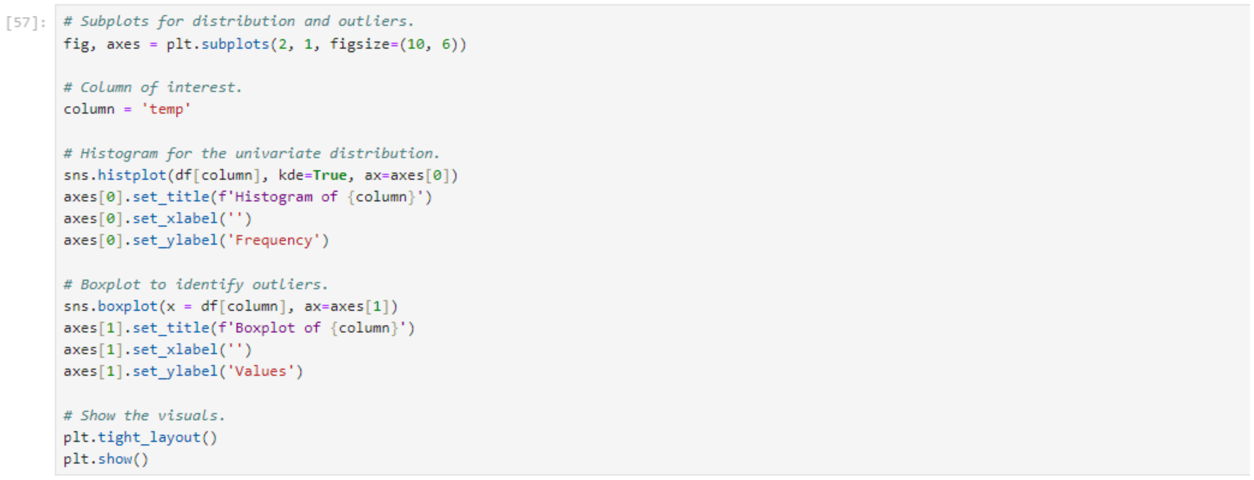
```
[12]: # Importing matplotlib.pyplot for Visualizations.
      import matplotlib.pyplot as plt

      # Import Seaborn to Build on matplotlib.pyplot.
      import seaborn as sns
```

Note. Importing libraries to visualize outliers, seaborn builds upon what is imported from the matplotlib library.

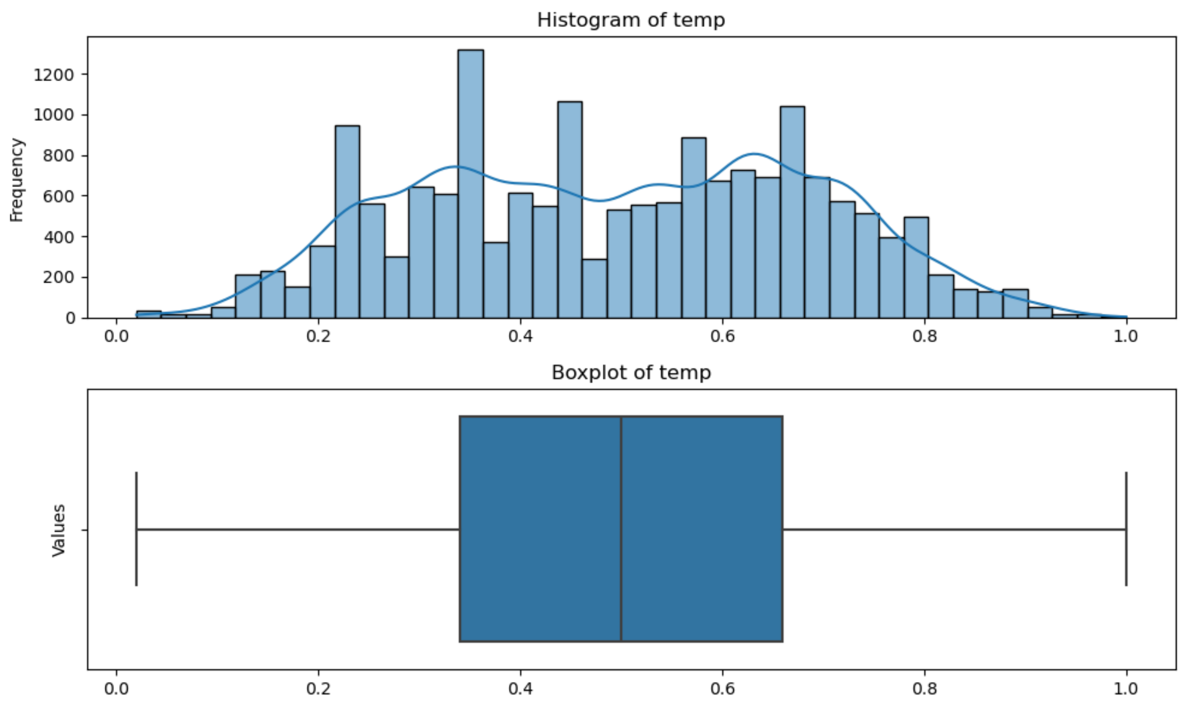
Since the dataset contains all values for all features provided, the following steps pertain to addressing outliers. To first address outliers, each continuous feature should be visualized along with its distribution in the dataset. Finding outliers will then require programmatic imputation, and the feature will be revisualized to show its cleaned nature.

Figure 6

Detecting outliers: temp



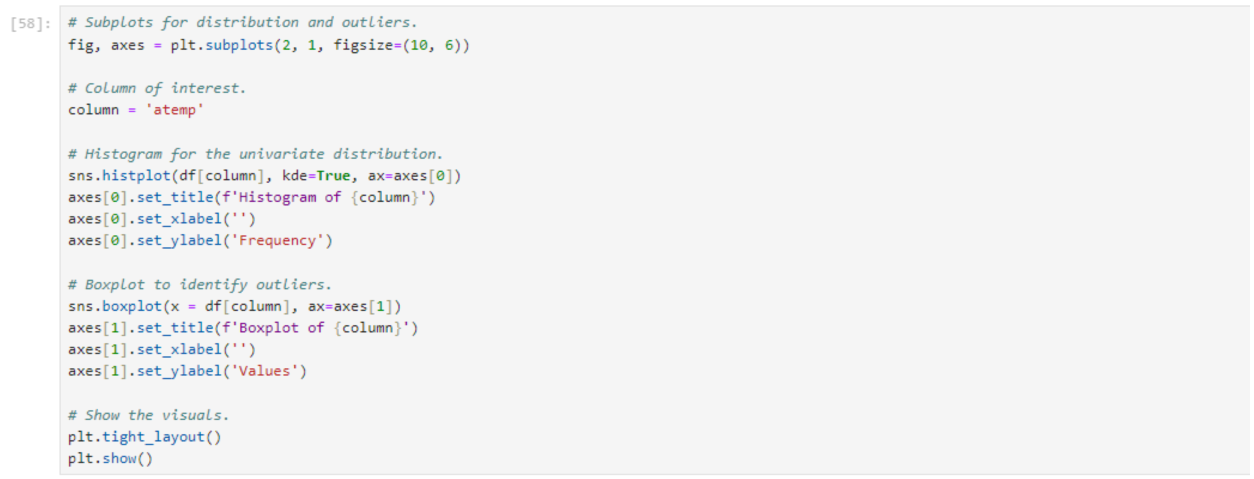
```
[57]: # Subplots for distribution and outliers.
      fig, axes = plt.subplots(2, 1, figsize=(10, 6))

      # Column of interest.
      column = 'temp'

      # Histogram for the univariate distribution.
      sns.histplot(df[column], kde=True, ax=axes[0])
      axes[0].set_title(f'Histogram of {column}')
      axes[0].set_xlabel('')
      axes[0].set_ylabel('Frequency')

      # Boxplot to identify outliers.
      sns.boxplot(x = df[column], ax=axes[1])
      axes[1].set_title(f'Boxplot of {column}')
      axes[1].set_xlabel('')
      axes[1].set_ylabel('Values')

      # Show the visuals.
      plt.tight_layout()
      plt.show()
```



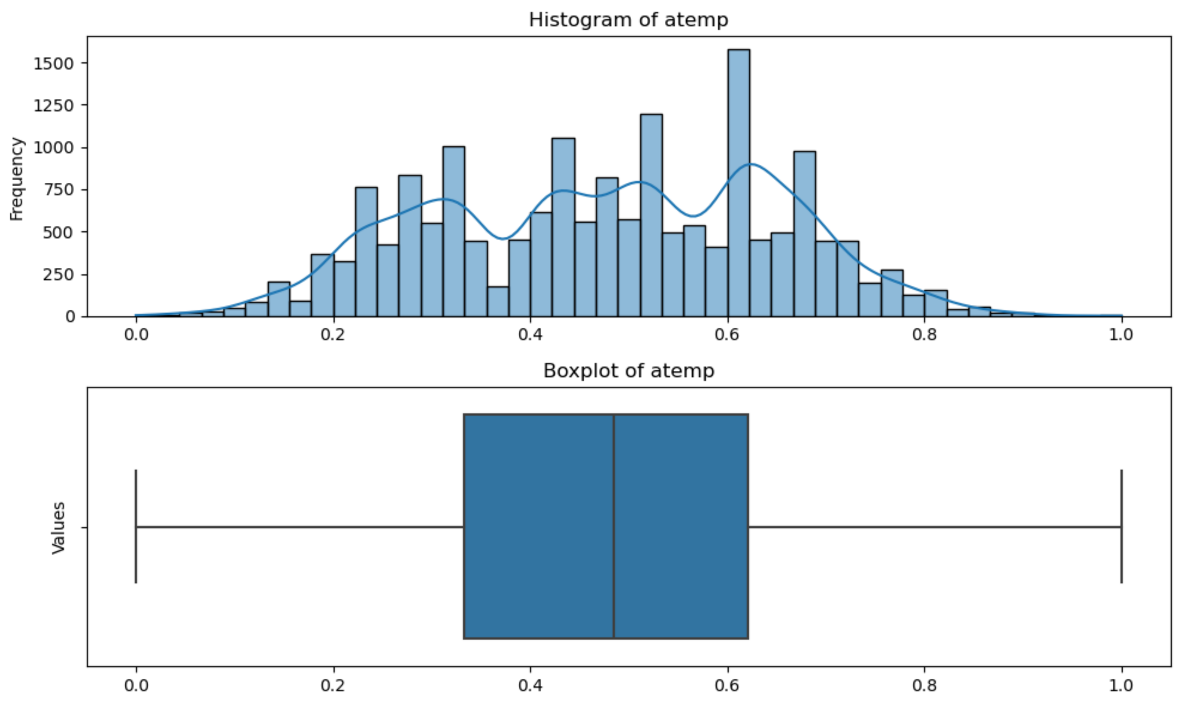
Note. Distribution and detection of outliers for the ‘temp’ variable. The variable ‘temp,’ as stated in the provided index file downloaded from UC Irvine’s Machine Learning Repository, is the temperature in Celsius divided by forty-one – Temperature (°C / 41).

Figure 7

Detecting outliers: atemp



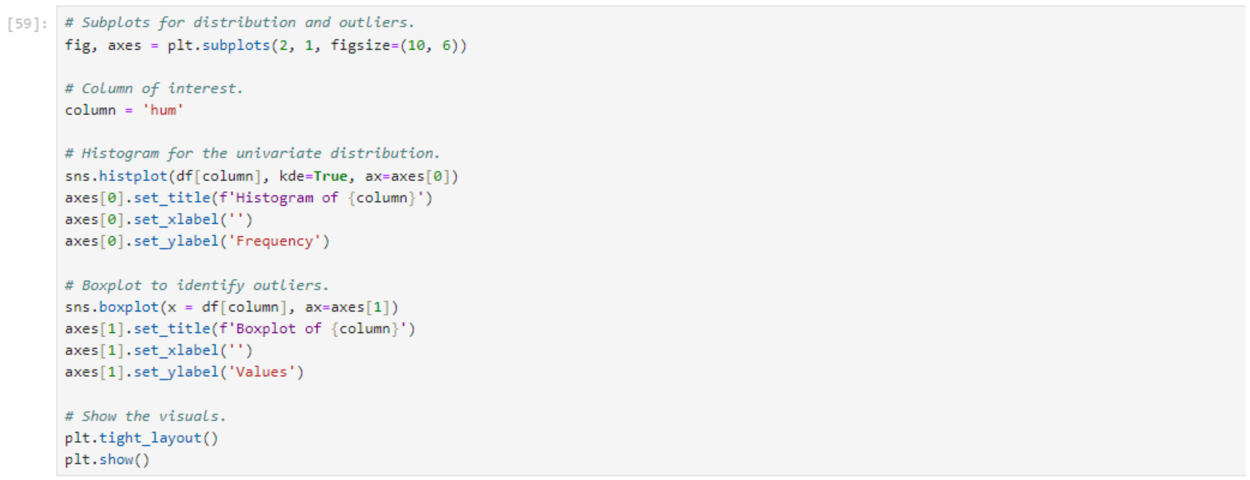
```
[58]: # Subplots for distribution and outliers.
      fig, axes = plt.subplots(2, 1, figsize=(10, 6))

      # Column of interest.
      column = 'atemp'

      # Histogram for the univariate distribution.
      sns.histplot(df[column], kde=True, ax=axes[0])
      axes[0].set_title(f'Histogram of {column}')
      axes[0].set_xlabel('')
      axes[0].set_ylabel('Frequency')

      # Boxplot to identify outliers.
      sns.boxplot(x = df[column], ax=axes[1])
      axes[1].set_title(f'Boxplot of {column}')
      axes[1].set_xlabel('')
      axes[1].set_ylabel('Values')

      # Show the visuals.
      plt.tight_layout()
      plt.show()
```



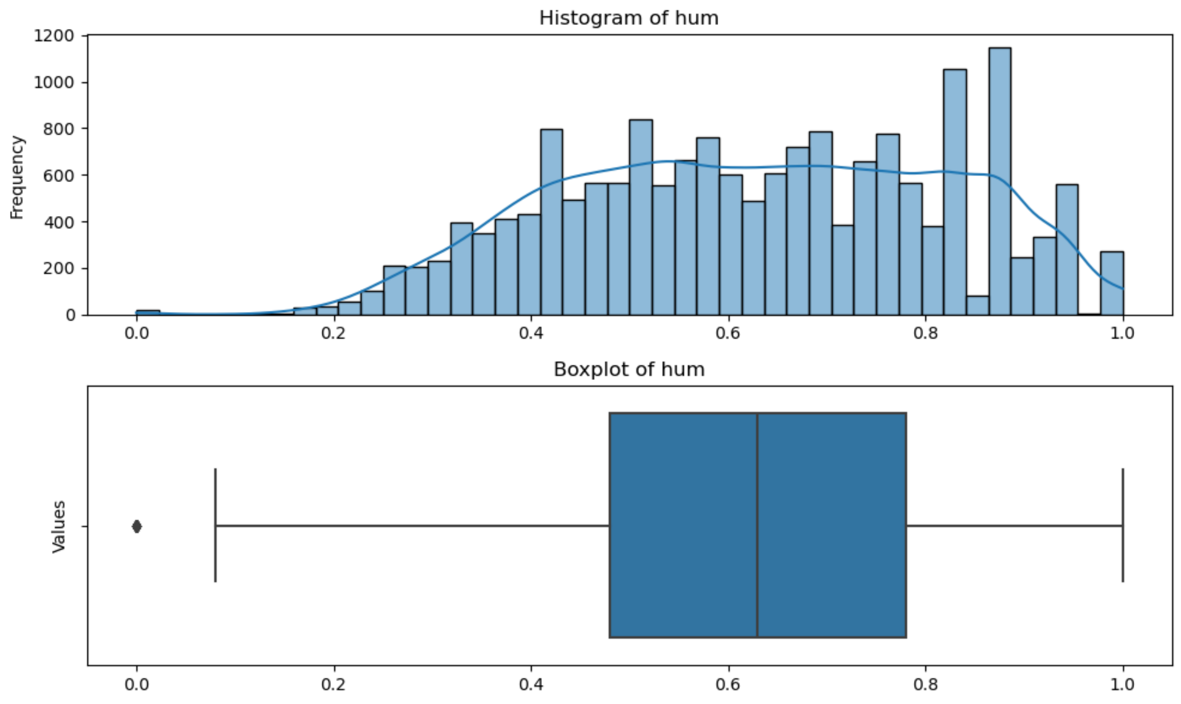
Note. Distribution and detection of outliers for the ‘atemp’ variable. The variable ‘atemp,’ as stated in the provided index file downloaded from UC Irvine’s Machine Learning Repository, is the feeling temperature in Celsius divided by fifty – Feeling temperature (^{o}C / 50).

Figure 8

Detecting outliers: hum



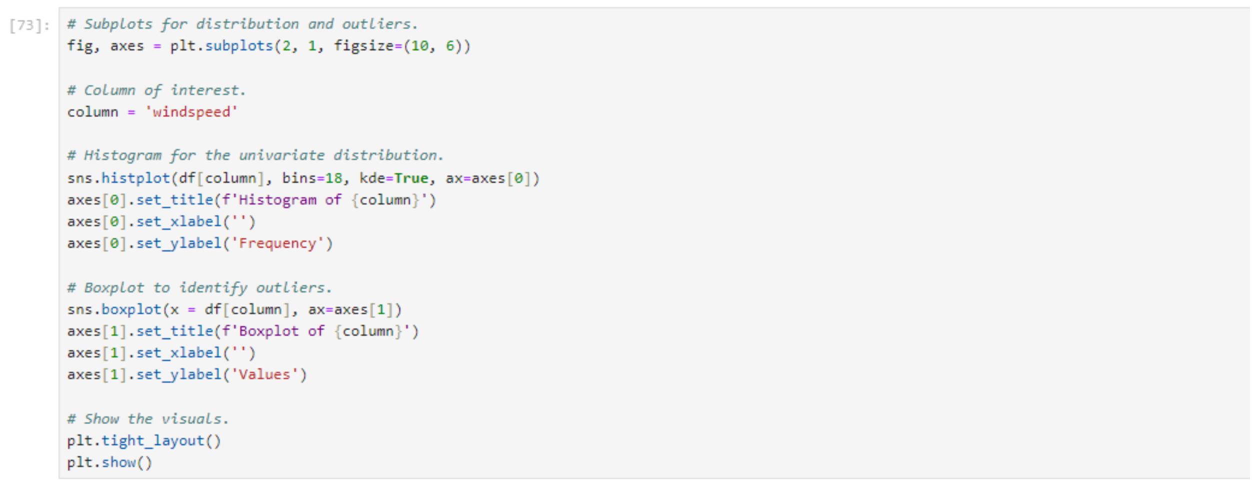
```
[59]: # Subplots for distribution and outliers.
      fig, axes = plt.subplots(2, 1, figsize=(10, 6))

      # Column of interest.
      column = 'hum'

      # Histogram for the univariate distribution.
      sns.histplot(df[column], kde=True, ax=axes[0])
      axes[0].set_title(f'Histogram of {column}')
      axes[0].set_xlabel('')
      axes[0].set_ylabel('Frequency')

      # Boxplot to identify outliers.
      sns.boxplot(x = df[column], ax=axes[1])
      axes[1].set_title(f'Boxplot of {column}')
      axes[1].set_xlabel('')
      axes[1].set_ylabel('Values')

      # Show the visuals.
      plt.tight_layout()
      plt.show()
```



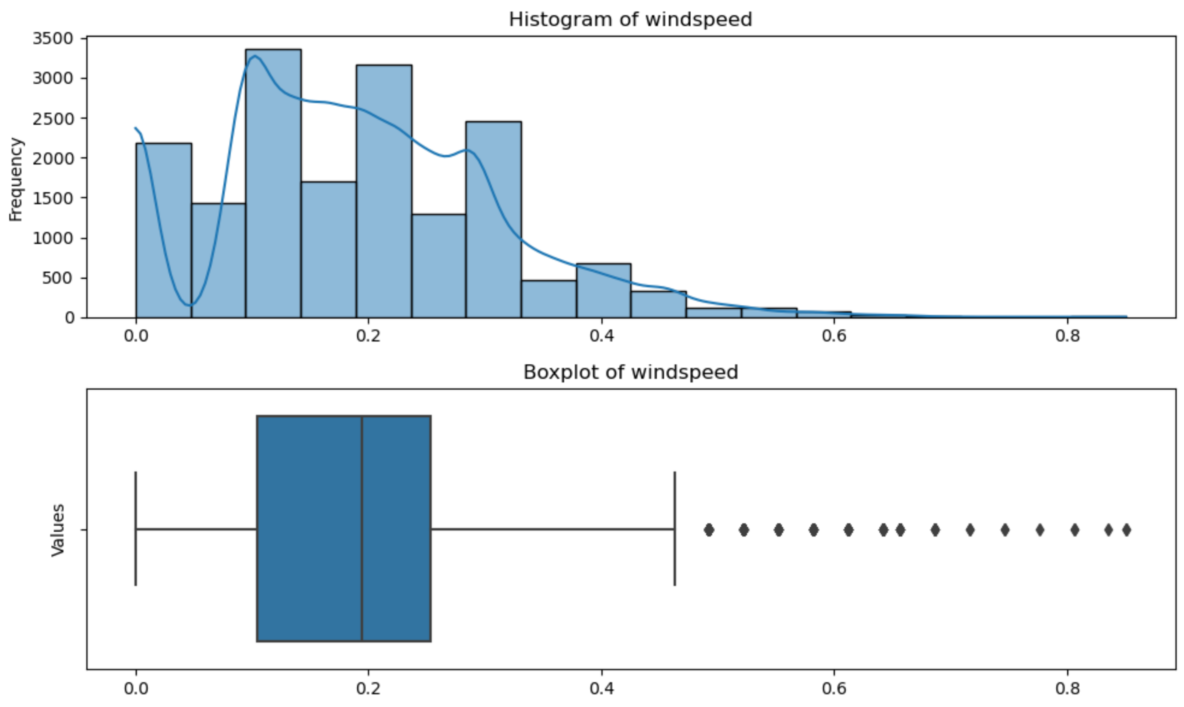
Note. Distribution and detection of outliers for ‘hum’ variable. The variable ‘hum,’ as stated in the provided index file downloaded from UC Irvine’s Machine Learning Repository, is the humidity factor divided by 100. This is *likely* represented as a percentage given the denominator – Humidity (value / 100).

Figure 9

Detecting outliers: windspeed



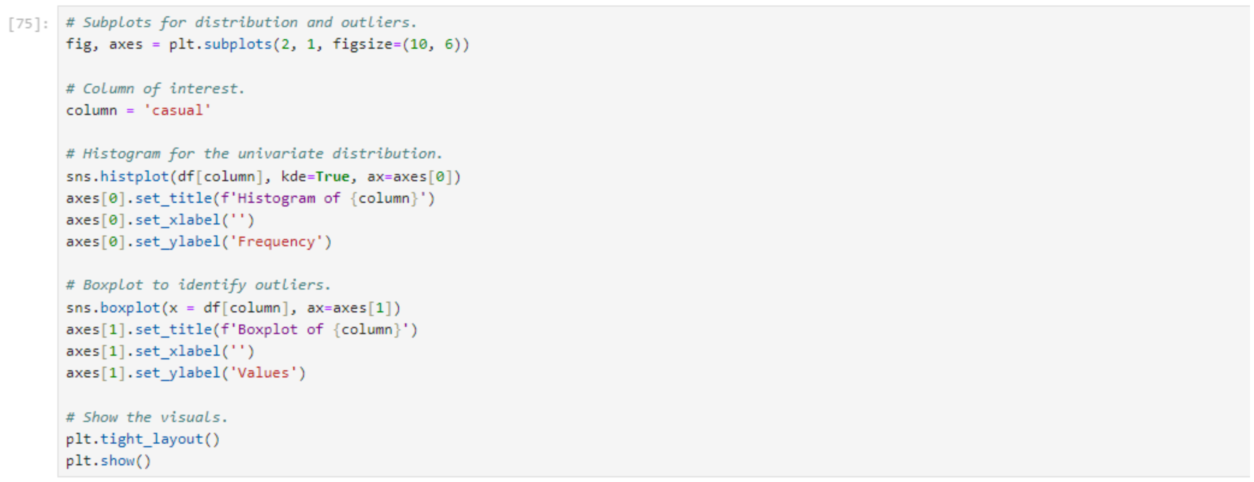
```
[73]: # Subplots for distribution and outliers.
      fig, axes = plt.subplots(2, 1, figsize=(10, 6))

      # Column of interest.
      column = 'windspeed'

      # Histogram for the univariate distribution.
      sns.histplot(df[column], bins=18, kde=True, ax=axes[0])
      axes[0].set_title(f'Histogram of {column}')
      axes[0].set_xlabel('')
      axes[0].set_ylabel('Frequency')

      # Boxplot to identify outliers.
      sns.boxplot(x = df[column], ax=axes[1])
      axes[1].set_title(f'Boxplot of {column}')
      axes[1].set_xlabel('')
      axes[1].set_ylabel('Values')

      # Show the visuals.
      plt.tight_layout()
      plt.show()
```



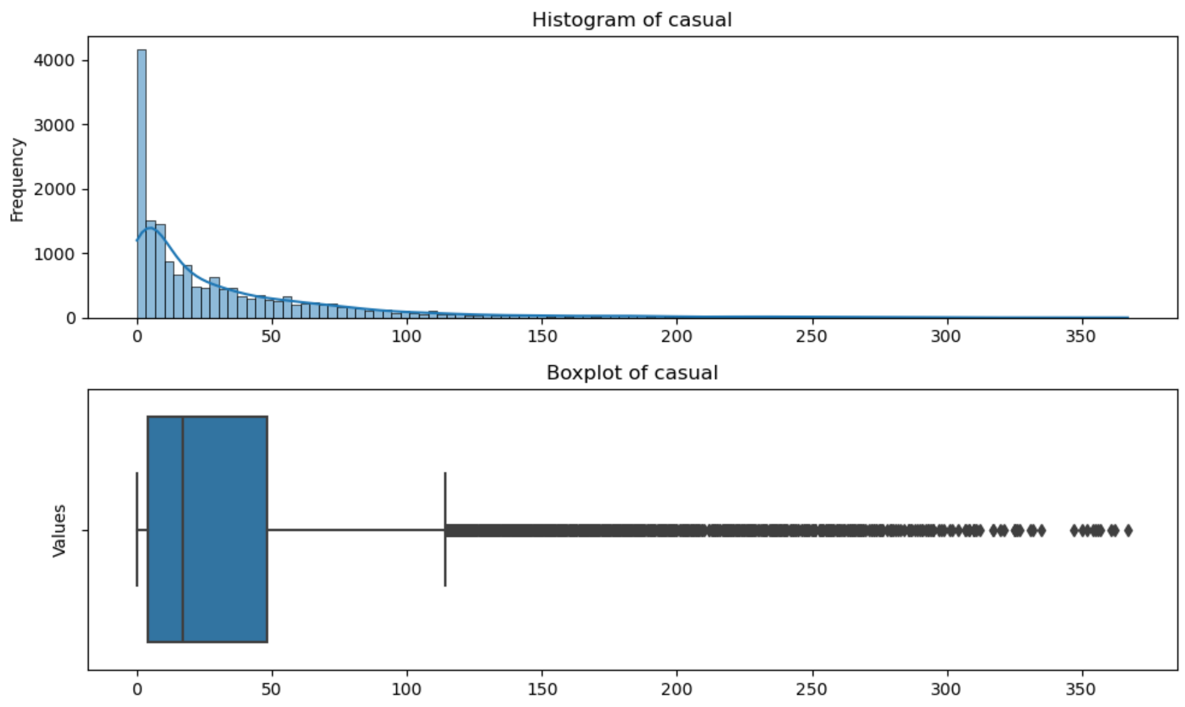
Note. Distribution and detection of outliers for ‘windspeed’ variable. There is no direct indication that this value is represented by a standard metrics. Since the dataset is representative of Washington D.C., the units are *likely* measured in miles per hour (mph) – Windspeed (mph / 67).

Figure 10

Detecting outliers: casual



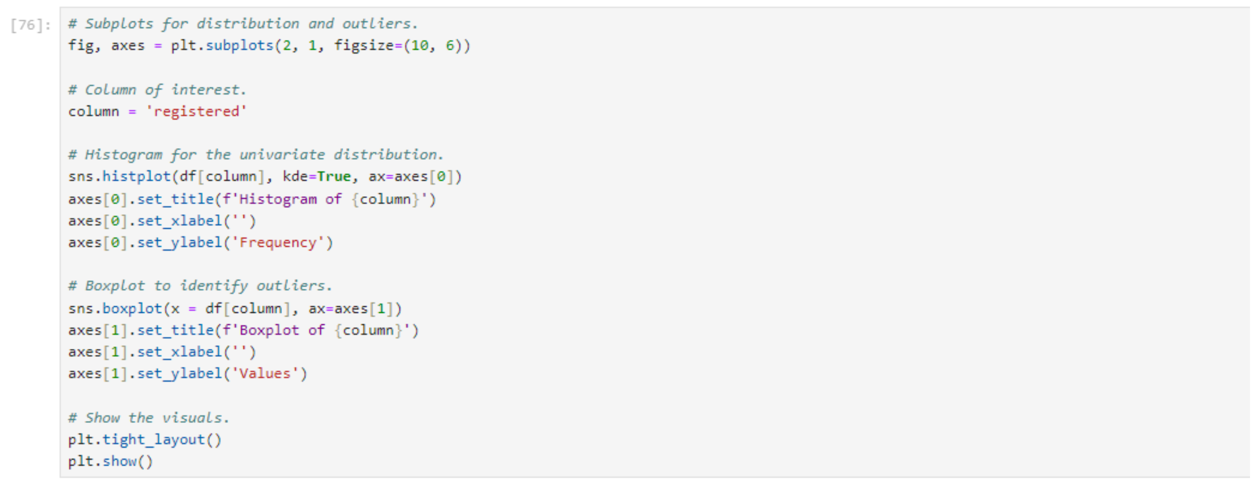
```
[75]: # Subplots for distribution and outliers.
      fig, axes = plt.subplots(2, 1, figsize=(10, 6))

      # Column of interest.
      column = 'casual'

      # Histogram for the univariate distribution.
      sns.histplot(df[column], kde=True, ax=axes[0])
      axes[0].set_title(f'Histogram of {column}')
      axes[0].set_xlabel('')
      axes[0].set_ylabel('Frequency')

      # Boxplot to identify outliers.
      sns.boxplot(x = df[column], ax=axes[1])
      axes[1].set_title(f'Boxplot of {column}')
      axes[1].set_xlabel('')
      axes[1].set_ylabel('Values')

      # Show the visuals.
      plt.tight_layout()
      plt.show()
```



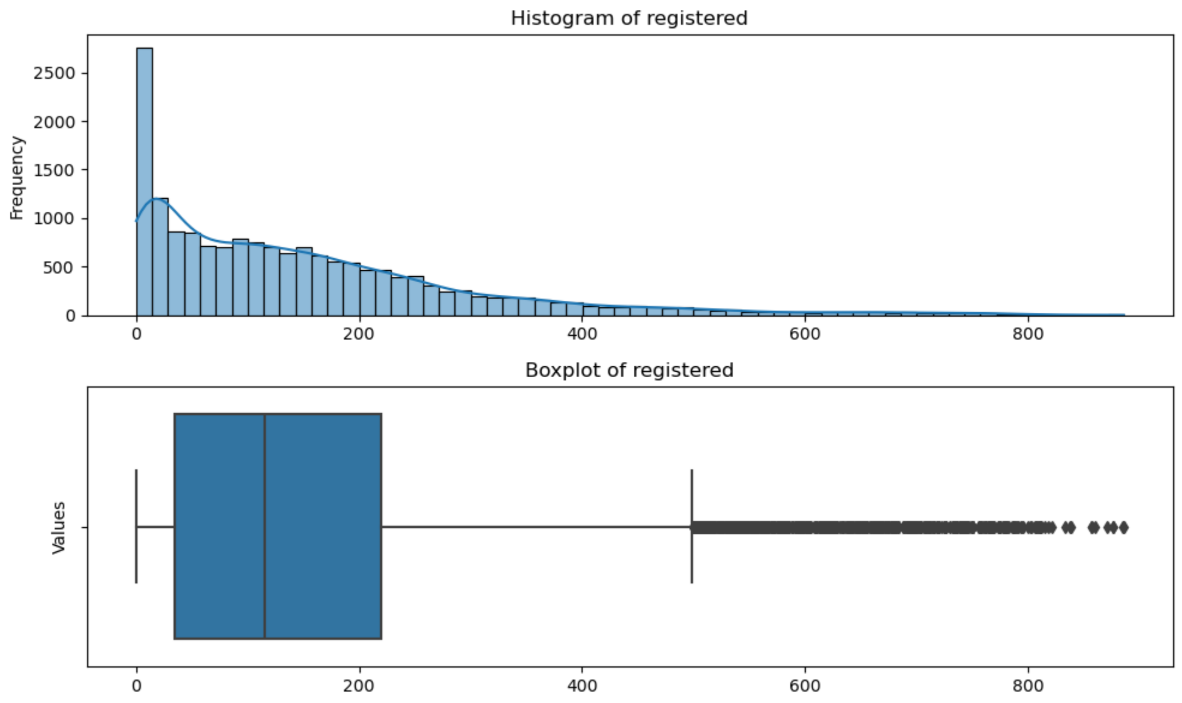
Note. Distribution and detection of outliers for ‘casual’ variable. The variable ‘casual,’ as stated in the provided index file downloaded from UC Irvine’s Machine Learning Repository is the count of casual users – Non-registered rentals.

Figure 11

Detecting outliers: registered



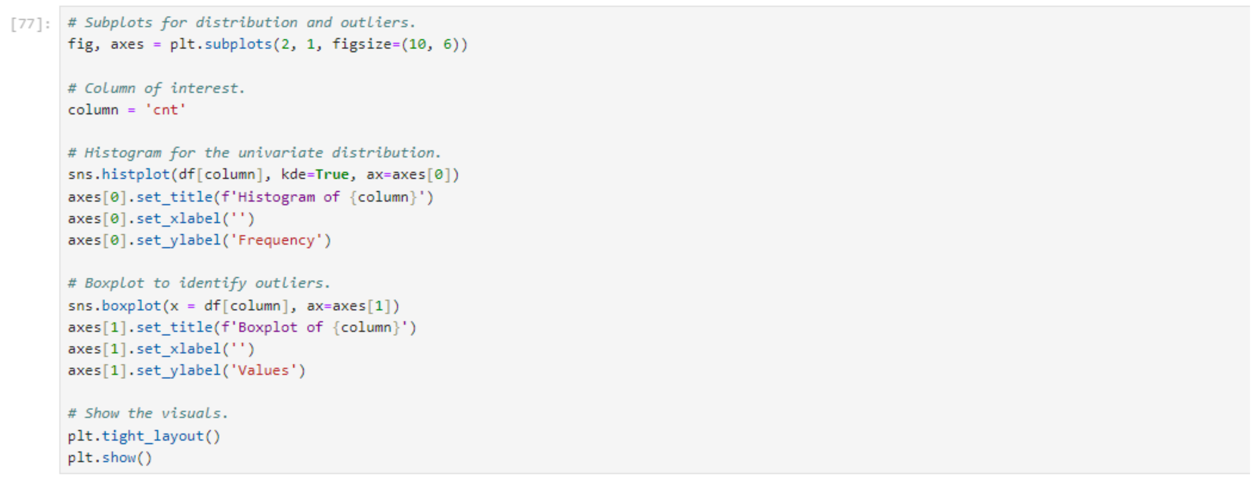
```
[76]: # Subplots for distribution and outliers.
fig, axes = plt.subplots(2, 1, figsize=(10, 6))

# Column of interest.
column = 'registered'

# Histogram for the univariate distribution.
sns.histplot(df[column], kde=True, ax=axes[0])
axes[0].set_title(f'Histogram of {column}')
axes[0].set_xlabel('')
axes[0].set_ylabel('Frequency')

# Boxplot to identify outliers.
sns.boxplot(x = df[column], ax=axes[1])
axes[1].set_title(f'Boxplot of {column}')
axes[1].set_xlabel('')
axes[1].set_ylabel('Values')

# Show the visuals.
plt.tight_layout()
plt.show()
```



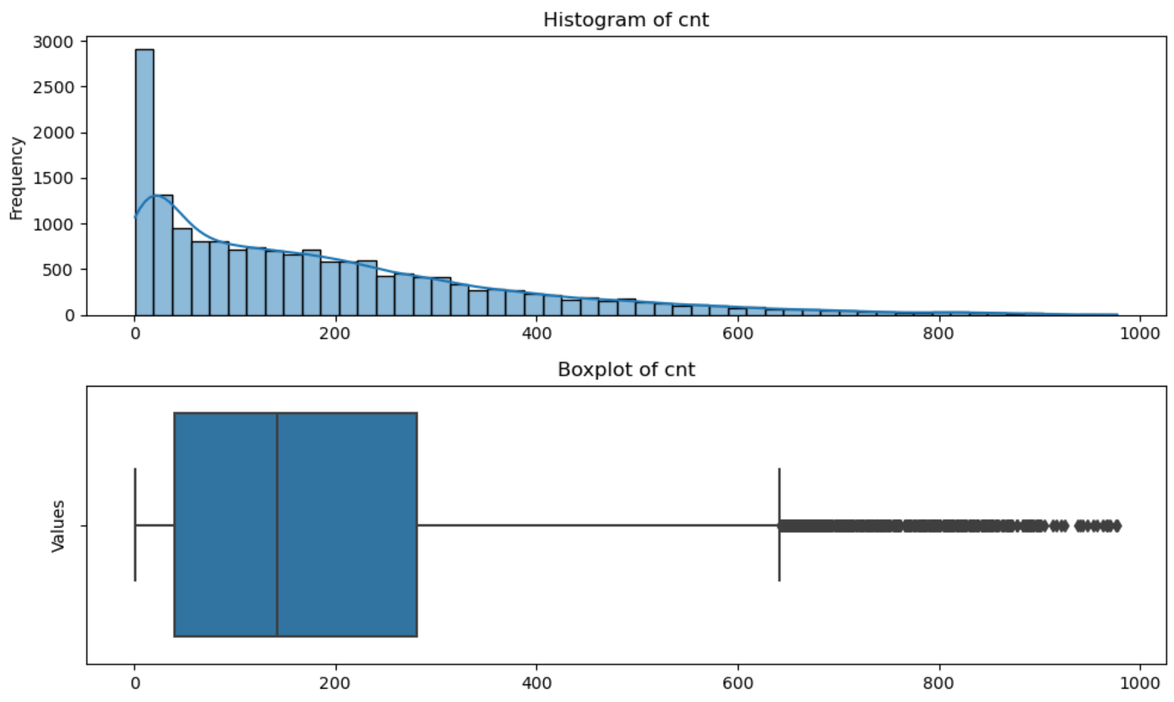
Note. Distribution and detection of outliers for ‘registered’ variable. The variable ‘registered,’ as stated in the provided index file downloaded from UC Irvine’s Machine Learning Repository is the count of registered users – Registered user rentals.

Figure 12

Detecting outliers: cnt



```
[77]: # Subplots for distribution and outliers.
      fig, axes = plt.subplots(2, 1, figsize=(10, 6))

      # Column of interest.
      column = 'cnt'

      # Histogram for the univariate distribution.
      sns.histplot(df[column], kde=True, ax=axes[0])
      axes[0].set_title(f'Histogram of {column}')
      axes[0].set_xlabel('')
      axes[0].set_ylabel('Frequency')

      # Boxplot to identify outliers.
      sns.boxplot(x = df[column], ax=axes[1])
      axes[1].set_title(f'Boxplot of {column}')
      axes[1].set_xlabel('')
      axes[1].set_ylabel('Values')

      # Show the visuals.
      plt.tight_layout()
      plt.show()
```

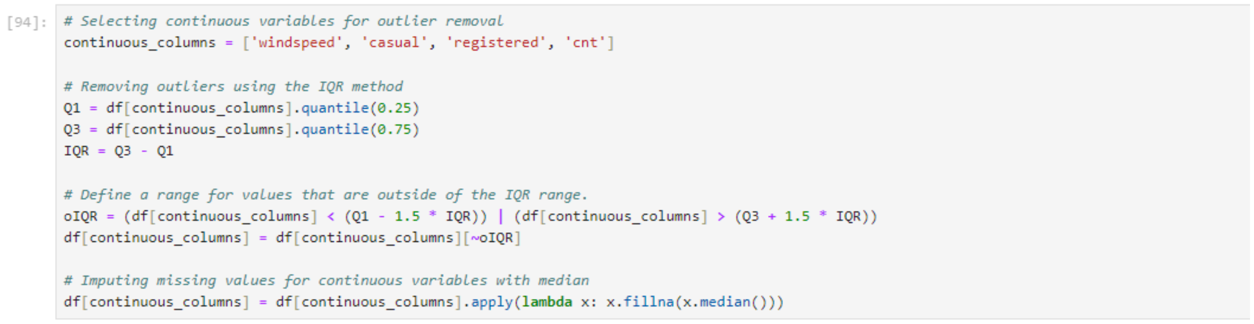


Note. Distribution and detection of outliers for ‘cnt’ variable. The variable ‘cnt,’ as stated in the provided index file downloaded from UC Irvine’s Machine Learning Repository is the count of bikes rented – Total count of bikes rented.

Outliers have been found in the columns ‘windspeed,’ ‘casual,’ ‘registered,’ and ‘cnt.’ In order to appropriately assess these outliers, they must be removed, and imputation must be performed to treat outliers in the analysis. The following figure, ‘*Programmatic removal and imputation of outliers,*’ demonstrates the programmatic removal and imputation of outliers, using the median for each feature.

Figure 13

Programmatic removal and imputation of outliers



```
[94]: # Selecting continuous variables for outlier removal
      continuous_columns = ['windspeed', 'casual', 'registered', 'cnt']

      # Removing outliers using the IQR method
      Q1 = df[continuous_columns].quantile(0.25)
      Q3 = df[continuous_columns].quantile(0.75)
      IQR = Q3 - Q1

      # Define a range for values that are outside of the IQR range.
      oIQR = (df[continuous_columns] < (Q1 - 1.5 * IQR)) | (df[continuous_columns] > (Q3 + 1.5 * IQR))
      df[continuous_columns] = df[continuous_columns][~oIQR]

      # Imputing missing values for continuous variables with median
      df[continuous_columns] = df[continuous_columns].apply(lambda x: x.fillna(x.median()))
```

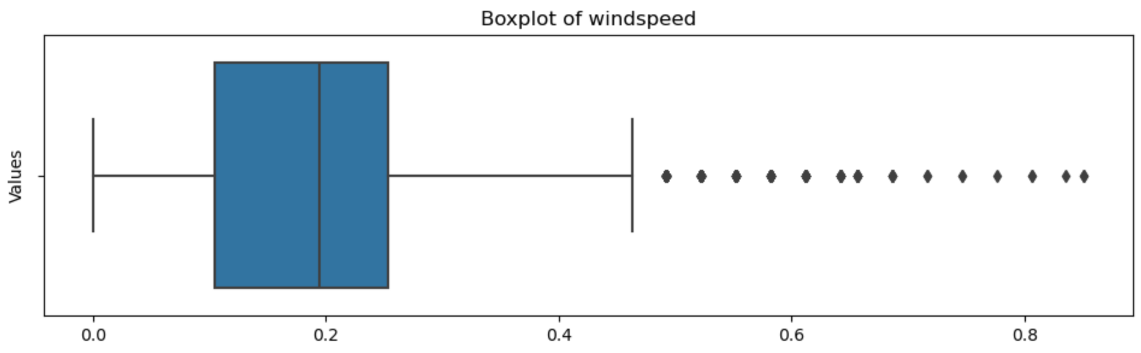
Note. Programmatic removal of outliers.

Now that the outliers have been removed and imputation has been performed, the update features will be displayed. It is important to note that outliers may still be present, but this time, they will stay to prevent overcleaning. By performing removal and imputation several times, the authenticity of the original dataset is lost. Support for this idea of why not to perform imputation more than once is found in Grace-Martin (2024).

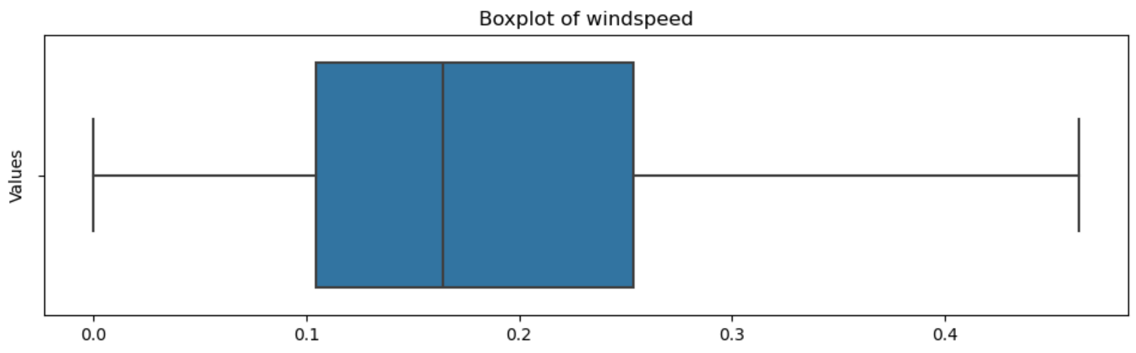
Figure 14

Imputation: windspeed

1. **Before**



2. **After**

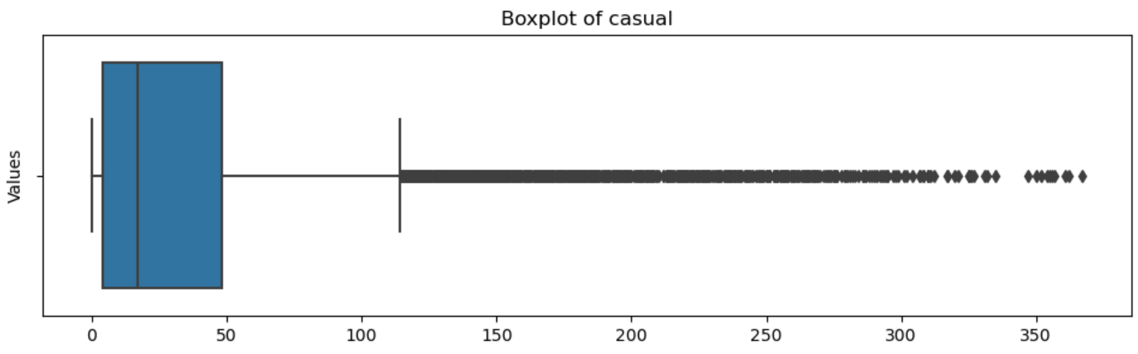


Note. Before and after results of imputation to the ‘windspeed’ independent variable.

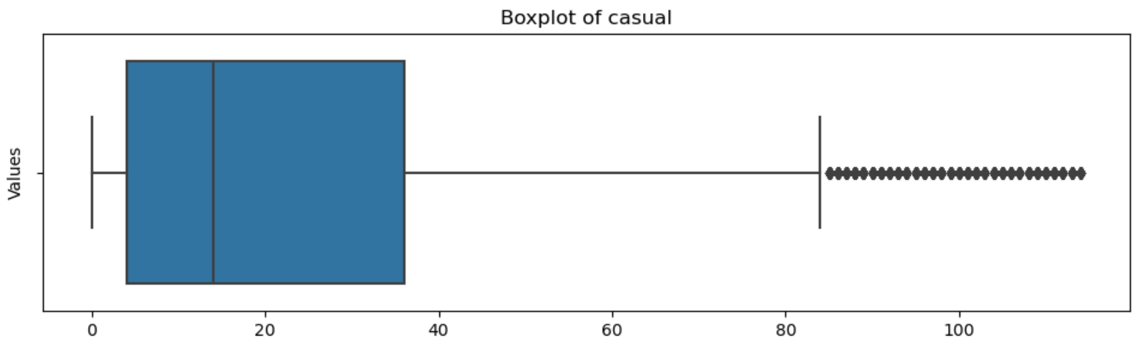
Figure 15

Imputation: casual

1. **Before**



2. **After**

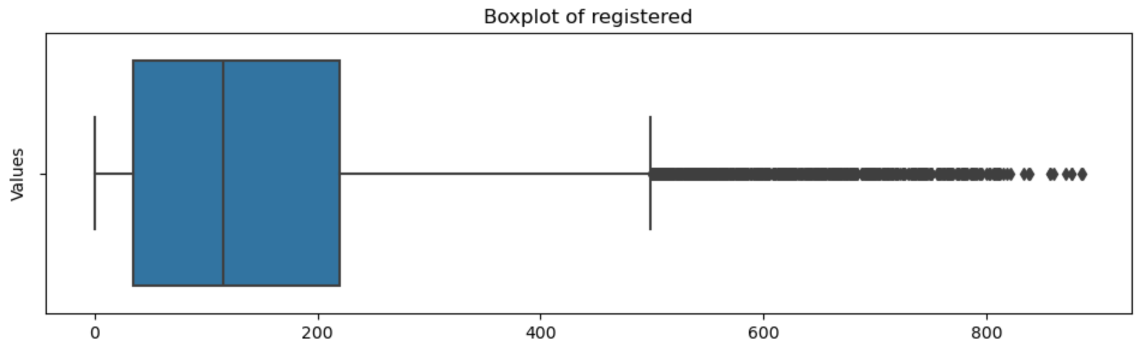


Note. Before and after results of imputation to the ‘casual’ independent variable.

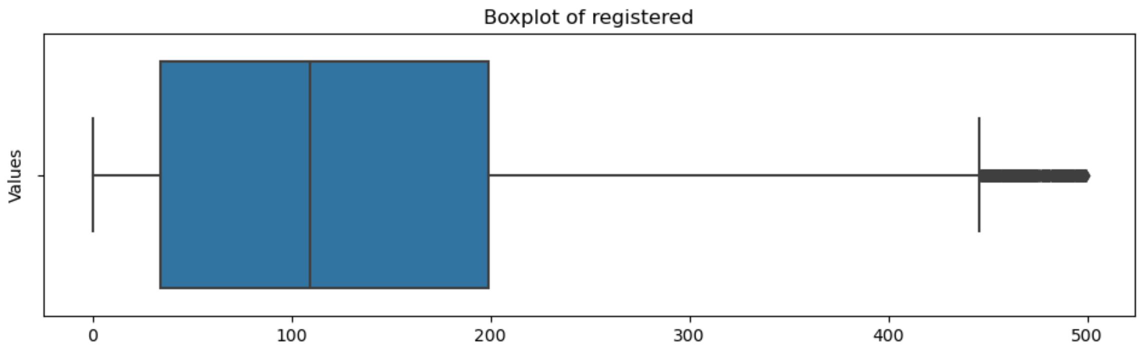
Figure 16

Imputation: registered

1. **Before**



2. **After**

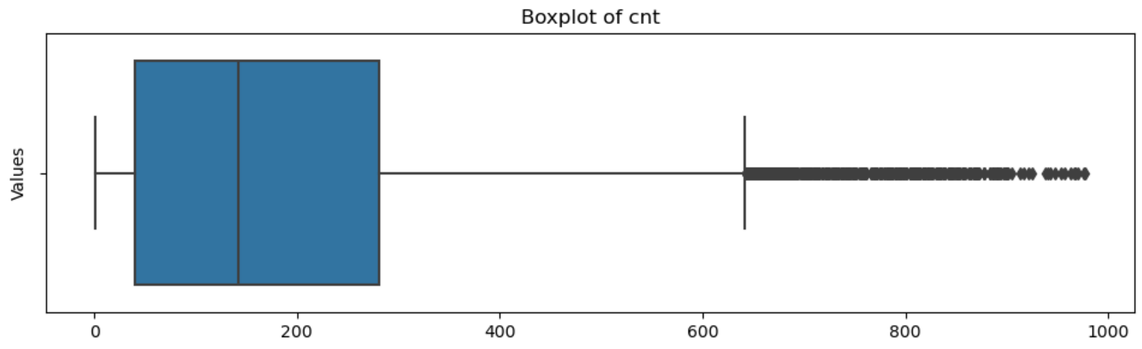


Note. Before and after results of imputation to the ‘registered’ independent variable.

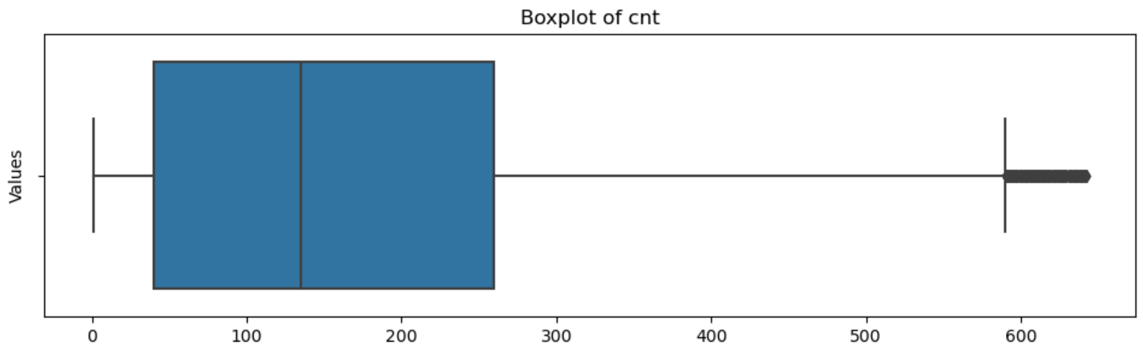
Figure 17

Imputation: cnt

1. **Before**



2. **After**

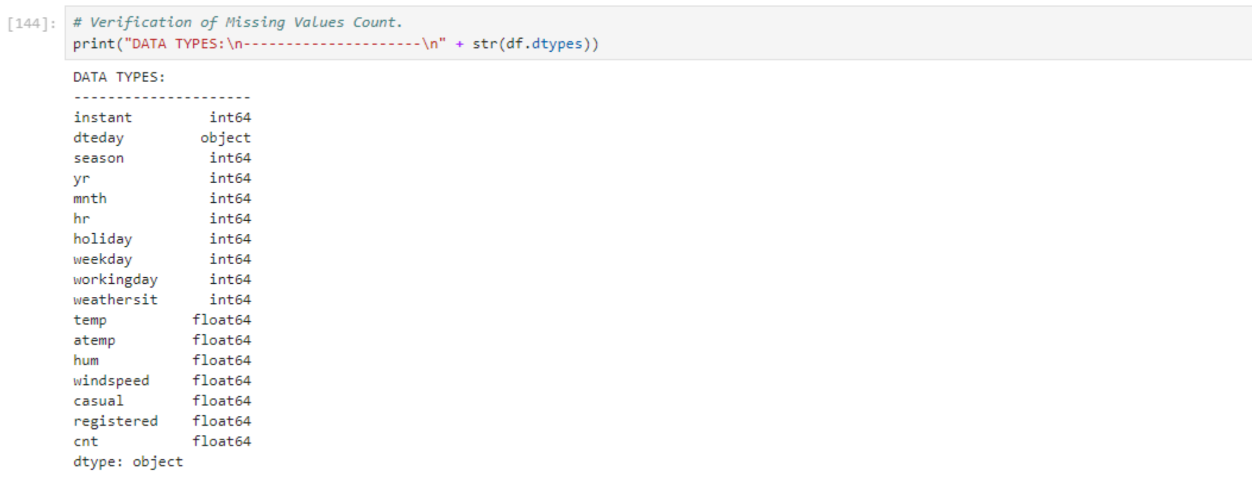


Note. Before and after results of imputation to the ‘cnt’ dependent variable.

Before model analysis, the data types must be appropriately converted into numerical format as required by most, if not all, linear regression models where implicit translation is not guaranteed. For example, this refers to converting boolean values, true or false, to numerical representations 1 or 0, respectively.

Figure 18

Data type conversion for model analysis.



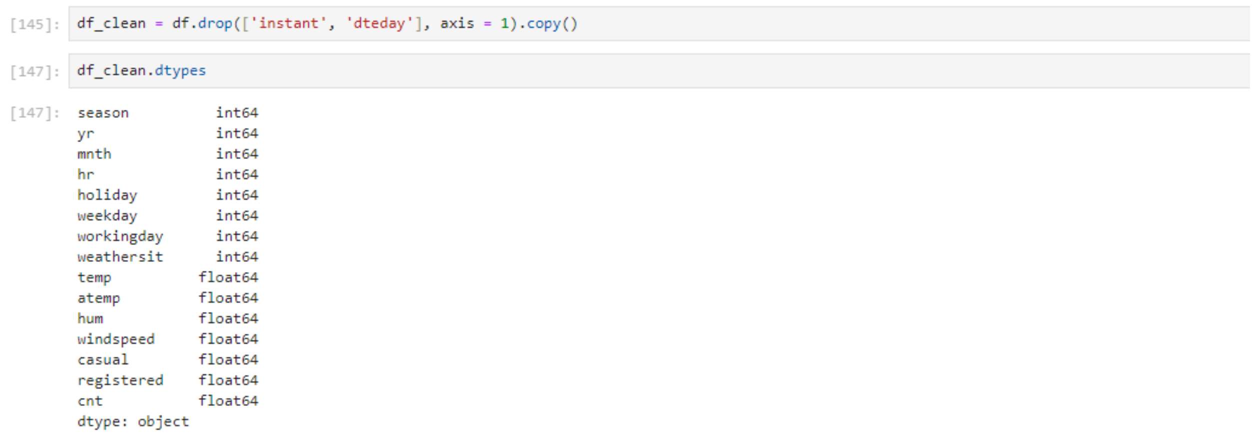
```
[144]: # Verification of Missing Values Count.
       print("DATA TYPES:\n--------------------\n" + str(df.dtypes))

       DATA TYPES:
       --------------------
       instant         int64
       dteday         object
       season          int64
       yr              int64
       mnth            int64
       hr              int64
       holiday         int64
       weekday         int64
       workingday      int64
       weathersit      int64
       temp          float64
       atemp         float64
       hum           float64
       windspeed     float64
       casual        float64
       registered    float64
       cnt           float64
       dtype: object
```

Note. The data types of each variable in Python. The data types are appropriate for the analysis without the need of conversion.

Pandas has already appropriately assigned integer or float based datatype format to all columns requiring such. It is important to note that ‘dteday’ is denoted as the *object* datatype introducing the next step of data preparation, removing unneccesary variables. Since variables like ‘mnth’ and ‘yr’ already exist in an ordinal format, the date is not required for this analysis and can be removed from the DataFrame, also using Pandas

Figure 19

Cleaned DataFrame, removing unnecessary variables.



```
[145]: df_clean = df.drop(['instant', 'dteday'], axis = 1).copy()
```

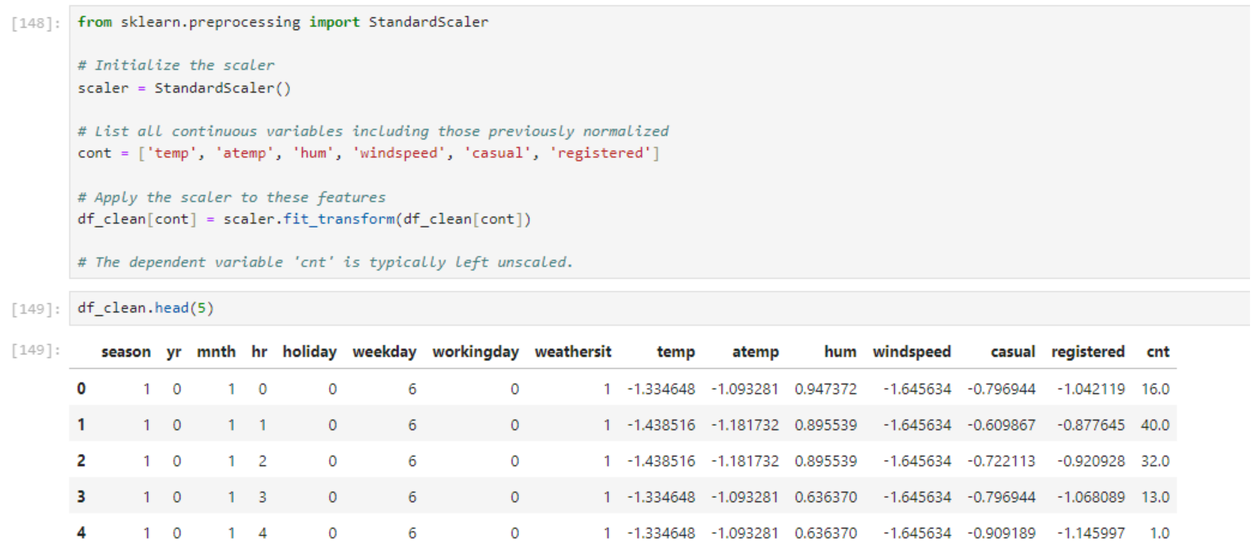
```
[147]: df_clean.dtypes
```

```
[147]: season          int64
       yr              int64
       mnth            int64
       hr              int64
       holiday         int64
       weekday         int64
       workingday      int64
       weathersit      int64
       temp          float64
       atemp         float64
       hum           float64
       windspeed     float64
       casual        float64
       registered    float64
       cnt           float64
       dtype: object
```

Note. A new DataFrame, copying the old DataFrame by removing the unneccessary variables has been created as *df_clean*.

Figure 20

Standardizing independent variables.



```
[148]: from sklearn.preprocessing import StandardScaler

       # Initialize the scaler
       scaler = StandardScaler()

       # List all continuous variables including those previously normalized
       cont = ['temp', 'atemp', 'hum', 'windspeed', 'casual', 'registered']

       # Apply the scaler to these features
       df_clean[cont] = scaler.fit_transform(df_clean[cont])

       # The dependent variable 'cnt' is typically left unscaled.
```

```
[149]: df_clean.head(5)
```

[149]:

	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	0	1	0	0	6	0	1	-1.334648	-1.093281	0.947372	-1.645634	-0.796944	-1.042119	16.0
1	1	0	1	1	0	6	0	1	-1.438516	-1.181732	0.895539	-1.645634	-0.609867	-0.877645	40.0
2	1	0	1	2	0	6	0	1	-1.438516	-1.181732	0.895539	-1.645634	-0.722113	-0.920928	32.0
3	1	0	1	3	0	6	0	1	-1.334648	-1.093281	0.636370	-1.645634	-0.796944	-1.068089	13.0
4	1	0	1	4	0	6	0	1	-1.334648	-1.093281	0.636370	-1.645634	-0.909189	-1.145997	1.0

Note. The continuous variables have been scaled to ensure equal basis for the regression and appropriate equation.

Justification of Tools and Techniques

The data extraction and preparation tools were libraries native to Python, a data-centric programming language. First, the Pandas library extracts the data from the CSV file downloaded from UC Irvine’s Machine Learning Repository, ‘hour.csv.’ Pandas provides an in-memory data structure to later manipulate/prepare from the data stored in a CSV file using Pandas’ CSV reading function. By having an in-memory dataset, an MLR model can be created once the dataset is prepared for analysis. Pandas is also used to assess the count of missing values, to which there were no missing values in the dataset. The data types of the variables in the dataset should be appropriate for model analysis and thus require to be converted to float/integer format, even for categorical variables that are ordinal, nominal, and even binary where true is denoted by one and false, 0. Lastly, Pandas assists in creating a cleaned dataset by providing tools to drop the unnecessary or inappropriate variables for analysis.

Second, missing values and outliers must be assessed and treated to prepare the dataset appropriately. Since missing values were previously assessed using Pandas’, in order to assess the dataset for outliers, matplotlib works in tandem with Pandas’ DataFrames in order to visualize outliers with a box-and-whisker plot which indicates what variables need treatment for outliers. Since the four variables, ‘windspeed,’ ‘casual,’ ‘registered,’ and ‘cnt,’ all have outliers and can be verified visually, matplotlib and seaborn are critical tools for this analysis. Seaborn builds on matplotlib for more accessible programming and better visualizations (Karl, 2024).

Scikit-learn’s StandardScaler is required as all variables should have an equal basis without any independent variable influencing the analysis/model. For example, even though most continuous variables were already *normalized*, the independent variables, such as casual rentals or registered user rentals, which are counts, are significantly larger than the normalized features. The rentals that are not the dependent variable are considered essential for the analysis when attempting to explain the data variance and thus require scaling appropriate with the other variables in the analysis.

Advantage

An advantage of the data extraction and preparation tools is the simple yet robust visualizations from matplotlib and Seaborn. By visualizing the data extraction and preparation process, these tools provide an easy method to see what has been done to the dataset to ensure proper extraction and preparation. As stated, ‘Overall, both libraries have their strengths and weaknesses’ (Karl, 2024), and both are great tools for visualizing data in any context, like data extraction and preparation for linear regression.

Disadvantage

A disadvantage of the data extraction and preparation tools used is the performance limitations of these libraries. For instance, Pandas operates in memory, meaning that the specs of the host machine limit the size of the data it can handle. As the dataset grows, processing it can become inefficient or impossible. The dataset used for this study consists of more than 15,000 rows. It is not exponentially large, but it is not a tiny dataset either.

Analysis

Data Analysis Process

The goal of this study and what is expected to be found is that the independent variables will be predictive enough so the model metrics, such as R-squared, will be 60% or greater, as seen with Kashyap (2021) and that the model is supportable for other bike rental venues. In order to create the model, the data requires separation into training and testing sets based on the independent and dependent variables utilizing scikit-learn's 'train_test_split' function.

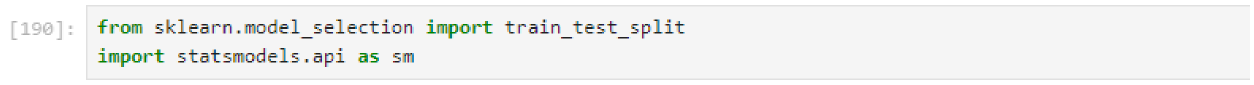
After the data is split into the appropriate training and testing sets, the model will be created by importing 'statsmodels.api'. First, a constant will be added to the independent variables to include an intercept in the model. Next, as the data is fit and predictions are made using statsmodels.api function calls for OLS, scores such as R-squared (R2) and Mean Squared Error (MSE) measure the model's accuracy. For this study, the aim is to create a model with an R2 score greater than or equal to 60%. Additionally, the statistical significance of each variable is evaluated, and a detailed linear regression equation is created to share the effect of each independent variable on the dependent variable.

Determining the normality for linear regression is vital for validating the model's assumption regarding the scedasticity (i.e., with the variance of error terms). A Quantile-Quantile plot (Q-Q plot) compares the residuals' quantiles to the normal distribution. If the residuals are normally distributed, an assumption of linear regression (Frost, 2018), the points will lie similar to a 45-degree line on the Q-Q plot, thus validating the assumption for this study.

Calculations and Outputs

Figure 21

Necessary Python imports.

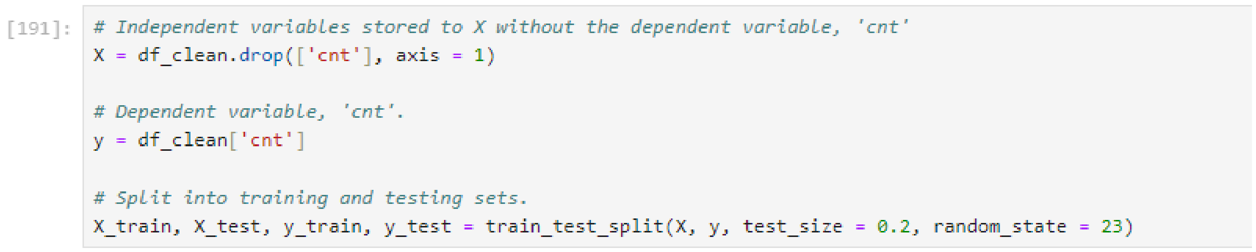


```
[190]: from sklearn.model_selection import train_test_split
       import statsmodels.api as sm
```

Note. Scikit-learn (sklearn) and Statsmodels (sm) are used for splitting the data into training and testing sets, and model creation, respectively.

Figure 22

Training and testing sets.

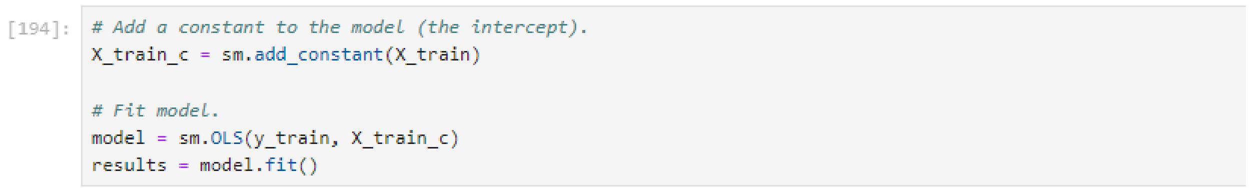


```
[191]: # Independent variables stored to X without the dependent variable, 'cnt'
       X = df_clean.drop(['cnt'], axis = 1)

       # Dependent variable, 'cnt'.
       y = df_clean['cnt']

       # Split into training and testing sets.
       X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 23)
```

Note. The data in the DataFrame is split into two classes and four total sets, a training and test set for the independent and dependent variables.

Figure 23

Constant and fit the model.

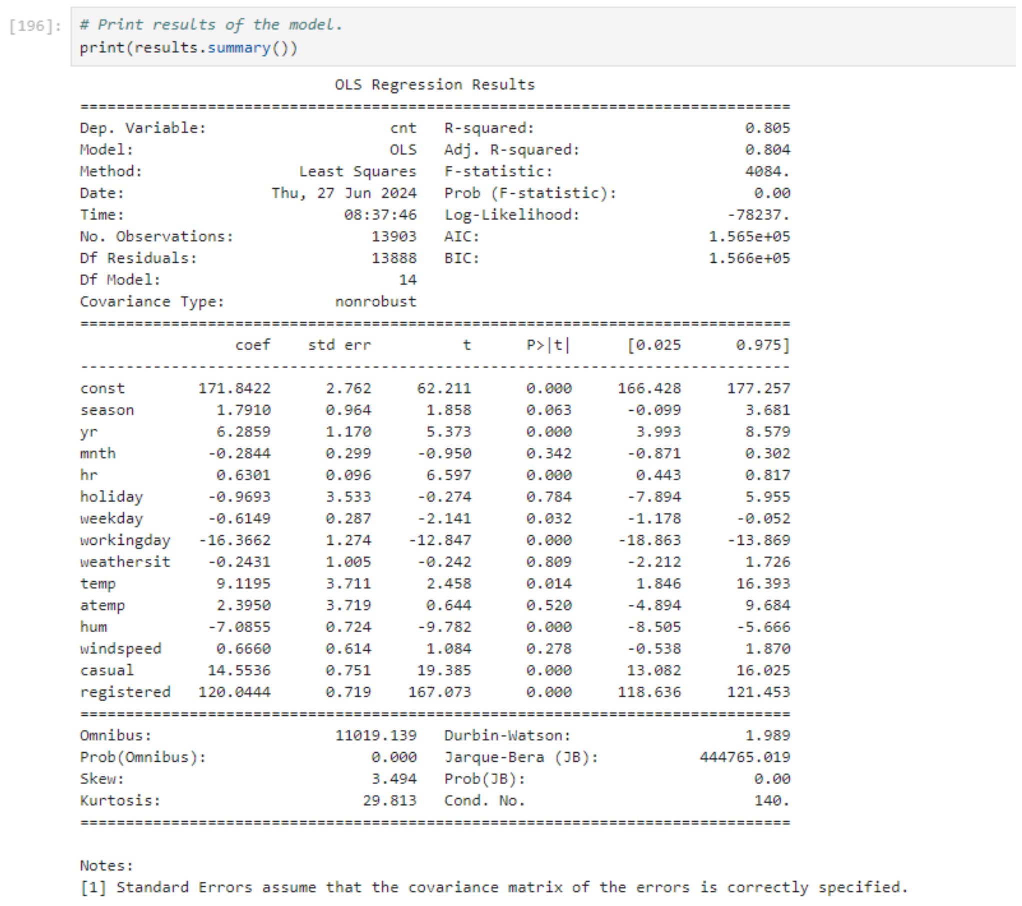


```
[194]: # Add a constant to the model (the intercept).
       X_train_c = sm.add_constant(X_train)

       # Fit model.
       model = sm.OLS(y_train, X_train_c)
       results = model.fit()
```

Note. The model and results are stored in respective variables.

Figure 24

Initial model summary.



```
[196]: # Print results of the model.
       print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.805
Model:                            OLS   Adj. R-squared:                  0.804
Method:                 Least Squares   F-statistic:                     4084.
Date:                Thu, 27 Jun 2024   Prob (F-statistic):               0.00
Time:                        08:37:46   Log-Likelihood:                -78237.
No. Observations:               13903   AIC:                         1.565e+05
Df Residuals:                   13888   BIC:                         1.566e+05
Df Model:                          14
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        171.8422      2.762     62.211      0.000     166.428     177.257
season         1.7910      0.964      1.858      0.063      -0.099       3.681
yr             6.2859      1.170      5.373      0.000       3.993       8.579
mnth          -0.2844      0.299     -0.950      0.342      -0.871       0.302
hr             0.6301      0.096      6.597      0.000       0.443       0.817
holiday       -0.9693      3.533     -0.274      0.784      -7.894       5.955
weekday       -0.6149      0.287     -2.141      0.032      -1.178      -0.052
workingday   -16.3662      1.274    -12.847      0.000     -18.863     -13.869
weathersit    -0.2431      1.005     -0.242      0.809      -2.212       1.726
temp           9.1195      3.711      2.458      0.014       1.846      16.393
atemp          2.3950      3.719      0.644      0.520      -4.894       9.684
hum           -7.0855      0.724     -9.782      0.000      -8.505      -5.666
windspeed      0.6660      0.614      1.084      0.278      -0.538       1.870
casual        14.5536      0.751     19.385      0.000      13.082      16.025
registered   120.0444      0.719    167.073      0.000     118.636     121.453
==============================================================================
Omnibus:                    11019.139   Durbin-Watson:                   1.989
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           444765.019
Skew:                           3.494   Prob(JB):                         0.00
Kurtosis:                      29.813   Cond. No.                         140.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Note. The results of the first model before backwards elimination.

Significant variables.

x_n	$p > 0.05$	β_n
yr	*0.000*	6.2859
hr	*0.000*	0.6301
weekday	*0.032*	-0.6149
workingday	*0.000*	-16.3662
temp	*0.014*	9.1195
hum	*0.000*	-7.0855
casual	*0.000*	14.5536
registered	*0.000*	120.0444

const = 171.8422

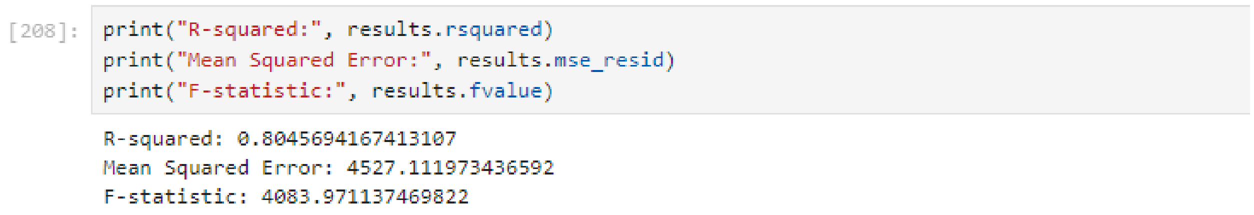
Regression equation.

Hourly count of bike rentals =

171.8422 + 6.2859 x (yr) + 0.6301 x (hr) – 0.6149 x (weekday) – 16.3662 x (workingday) +

9.1195 x (temp) – 7.0855 x (hum) + 14.5536 x (casual) + 120.0444 x (registered)

Accuracy metrics.

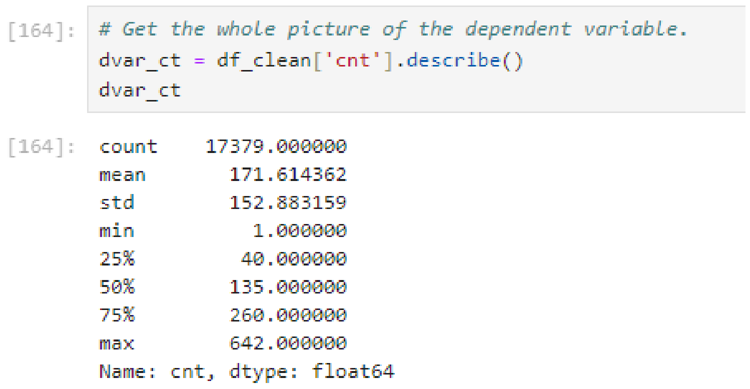


```
[208]: print("R-squared:", results.rsquared)
       print("Mean Squared Error:", results.mse_resid)
       print("F-statistic:", results.fvalue)

       R-squared: 0.8045694167413107
       Mean Squared Error: 4527.111973436592
       F-statistic: 4083.971137469822
```

Metric	Score	Significance
R^2 (R-squared)	*0.805*	
MSE (Mean Squared Error)	*4527.112*	
F-statistic	*4083.971*	p = *0.000*

Figure 25

Describing the dependent variable.

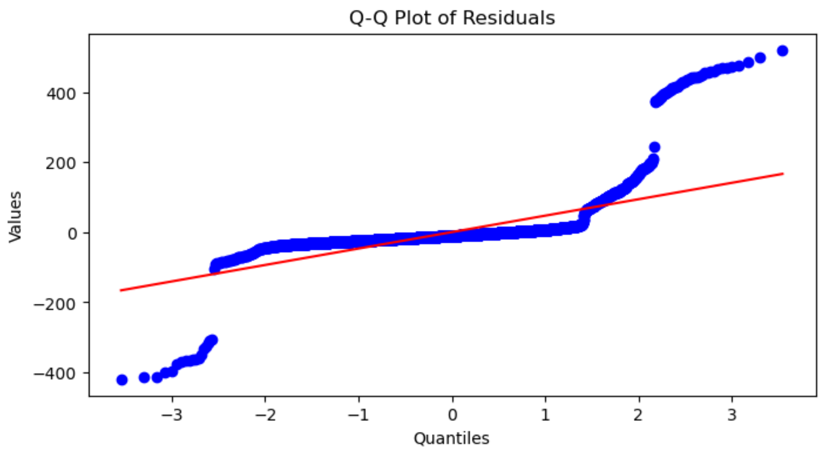


```
[164]: # Get the whole picture of the dependent variable.
       dvar_ct = df_clean['cnt'].describe()
       dvar_ct

[164]: count    17379.000000
       mean       171.614362
       std        152.883159
       min          1.000000
       25%         40.000000
       50%        135.000000
       75%        260.000000
       max        642.000000
       Name: cnt, dtype: float64
```

Note. Describing the dependent variable provides the mean and standard deviation, needed for context with MSE.

Figure 26

Quantile-Quantile plot.



Note. Quantile-Quantile plot for comparing the predicted values against the actual values to assess homoscedasticity, an assumption of linear regression.

Justification of Analysis Techniques

The data analysis process starts with using scikit-learn's train_test_split, which separates the data in the cleaned DataFrame into training and testing sets. Separate sets are required to validate the model based on data it has not seen. Splitting the data into training and testing sets is standard practice to confirm that the model will generalize the data it was trained on to reflect the observations in the dataset.

The model was then created using statsmodels.api and the associated OLS instance call to create a statistical model for the (multiple) linear regression analysis. The model added a constant to the independent variables to include an intercept that ensures the regression line appropriately fits the data it is modeled against.

Evaluation metrics such as R^2, MSE, and F-statistic assessed the model's accuracy and general performance. R^2 indicated that the variance in the dependent variable concerning the independent variables surpassed a threshold above 60% or greater. The MSE measured the average squared difference between the predicted and actual test values, known as residuals. The metrics for R^2 and MSE, used to measure the model's accuracy and fit, were 0.805 and 4527.112, respectively. The R^2 value suggests a strong model that supports the model's accuracy with a noticeable drawback from the MSE value. MSE value compares the mean and standard deviation of the dependent variable, which are considerably smaller than the MSE, suggesting that there are potentially considerable extreme values in the dataset's observations. Finally, the F-statistic assesses the null hypothesis in a manner that considers all independent variables *not* statistically significant. The overall model is statistically significant because the F-statistic score is large, and the associated p-value is significant at 0.000. This measure effectively rejects the null hypothesis.

The statistically significant independent variables were stored to construct a regression equation that generalizes the hourly count of bike rentals. The table keeps the coefficients or weights of each variable and the proof of statistical significance to justify how each of the standardized variables affects the dependent variable.

Lastly, a Q-Q plot is used to check the normality of the residuals, which is an assumption of linear regression for performing the prior statistical steps following homoscedasticity. The Q-Q plot verifies that the residuals approximately follow a normal distribution represented by values plotted close to the red line.

Advantage

The benefit of using linear regression and the techniques used is rooted in transparency. The detailed summaries for results, p-values, and more are essential in the modeling process. Transparency is vital for this application to validate and trust the model’s findings and predictions of the dependent variable. By offering the details behind each independent variable and other statistical measures such as R^2, MSE, F-statistic, and more, there is plenty of information to make comprehensive arguments for or against a model and why it is important in a practical application for market research.

Disadvantage

A disadvantage of linear regression and the techniques used is the sensitivity based on the assumptions of linear regression. For instance, outliers or extreme values, as seen in the Q-Q plot, can disproportionately influence the model, leading to biased or incorrect estimates, thus making a model unreliable. This can be seen minimally in the tail ends of the Q-Q plot in Figure 26, which affects the measures in the summary if the assumption of homoscedasticity is not met.

Data Summary and Implications

Implications and Results

The null hypothesis that a predictive MLR model cannot be constructed on the market research dataset, is rejected. The model achieved an R^2 score of 0.805, which implies that approximately 80.5% of the variability in the hourly bike rental counts can be explained by the model's independent variables. This score has significantly exceeded the initial target of 60% and thus is a strong predictive model. By passing the threshold of 60%, the model is statistically backed as a reliable tool for predicting bike rental count for the statistically significant variables.

The null hypothesis, H_0, that a predictive MLR model cannot be constructed, is rejected, while the alternate hypothesis, H_a, is supported. Ultimately, the approach and methods used in this study showcase the effectiveness of MLR in handling the variables seen to support a hypothesis based on the market research dataset.

The statistically significant variables of the model, 'yr,' 'hr,' 'weekday,' 'working day,' 'temp,' 'hum,' 'casual,' and 'registered,' indicate their impact on bike rental counts. Since their impact is available, given their coefficients, stakeholders are then given insight into understanding the factors that influence rental behavior, which may assist in planning and operational functionality.

The model provides a blueprint for other cities or businesses in the bike rental industry to predict demand based on similar variables or, at the surface level, the variables to consider. This may assist in inventory management, marketing strategy, and resource allocation to predict peak rental hours and adjust accordingly.

Limitation

A limitation of the analysis is recognized from the time range restriction of the dataset. The dataset is only for 2011 and 2012, which still provides valuable insight into how certain factors affect the total number of rental bikes in a given hour; it is from more than a decade ago. The dataset is also restricted to the Washington D.C. area, where relative factors like location and population differ globally (Eren & Uz, 2019), which would influence the use of rental bikes differently but are excluded from the analysis due to a lack of representation.

Recommended Course of Action

The model should be deployed to forecast bike rental demand to assist in managing inventory and staffing. Continuously updating the model with real-time data will keep predictions accurate and reflect current trends and changes in consumer behavior. Introducing the utility of the model also evaluates its effectiveness for different locales.

Expanding the dataset to include bike rental data from different locales helps find regional data to improve the model's ability to generalize. Utilizing the model also develops localized models for different regions to address specific environmental and seasonal factors. For instance, the average number of bike rentals based on casual and registered users varies by city.

Next, the model should change with time, meaning that new independent variables should be introduced to capture factors like marketing efforts, events, feedback, and more and how they affect rentals for independent variables that are not currently considered in the model. Specifically, variables like customer feedback can capture factors that have not been considered or removed from consideration.

Finally, continuously validate the model against new data and update it to reflect the ever-changing environmental and seasonal patterns to ensure its reliability and accuracy in the long term. Doing so also reveals where errors come from, helping to understand where the model fails and why.

Potential Directions

Similar to the recommended course of action, by expanding the geographical scope, the model's application would *likely* benefit from incorporating geographic location data to generalize better for a given city, state, etcetera. An additional direction is to explore different modeling techniques that are even more accurate than the current metrics measure. Exploring models like random forests has proven effective (Karunanithi et al., 2024), and other models like neural networks could potentially improve the predictive accuracy and, thus, effectivity.

References

Regression analysis: How do I interpret R-squared and assess the goodness-of-fit? (2021, March 15). Minitab Blog. https://blog.minitab.com/en/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit

Fanaee-T, H. (2013, December 19). *Bike Sharing*. UC Irvine Machine Learning Repository. https://archive.ics.uci.edu/dataset/275/bike+sharing+dataset

Eren, E., & Uz, V. E. (2019, October 15). *A review on bike-sharing: The factors affecting bike-sharing demand*. ScienceDirect. https://www.sciencedirect.com/science/article/abs/pii/S2210670719312387

Grace-Martin, K. (2024). *multiple imputation: 5 recent findings that change how to use it*. The Analysis Factor. https://www.theanalysisfactor.com/multiple-imputation-5-recent-findings-that-change-how-to-use-it/

Hitzek, J., Fischer-Rosinský, A., Möckel, M., Kuhlmann, S. L., & Slagman, A. (2022, April 25). *Influence of Weekday and Seasonal Trends on Urgency and In-hospital Mortality of Emergency Department Patients*. Frontiers. https://www.frontiersin.org/journals/public-health/articles/10.3389/fpubh.2022.711235/full

Jain, J., & Banka, P. (2024). *Dealing with outliers and missing values in a dataset*. NeenoPal. https://www.neenopal.com/dealing-with-outliers-and-missing-values-in-a-dataset.html

Karl, T. (2024, February 12). *How to choose between Seaborn vs. Matplotlib*. New Horizons. https://www.newhorizons.com/resources/blog/how-to-choose-between-seaborn-vs-matplotlib

Karunanithi, M., Chatasawapreeda, P., & Khan, T. A. (2024, May 22). *A predictive analytics approach for forecasting bike rental demand*. ScienceDirect. https://www.sciencedirect.com/science/article/pii/S2772662224000869#sec9

Kashyap, A. S. (2021). *Regression Model to Predict Bike Sharing Demand*. Academia. https://www.academia.edu/46510368/Regression_Model_to_Predict_Bike_Sharing_Demand?sm=b

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). *Scikit-Learn: Machine learning in Python*. Journal of Machine Learning Research. https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

Sahu, P. (2023, January 27). *A Comprehensive Guide to OLS Regression: Part-1*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2023/01/a-comprehensive-guide-to-ols-regression-part-1/